DepthCrafter: Generating Consistent Long Depth Sequences for Open-world Videos

Supplementary Material

Appendix

In this supplementary material, we provide additional implementation details in Appendix A and more evaluations in Appendix B. For more visual results and details, *we highly recommend referring to the webpage:* https://depthcrafter.github.io, since the visual quality of the generated depth sequences can be better accessed with interactive videos. We would release the code and model to facilitate further research and applications.

A. Implementation Details

A.1. Data Preparation

Following conventional practice in depth estimation [67, 68], we represent the depth in the disparity domain. We target relative depth estimation, so we normalize the disparity values to the range of $[0, 1]$ by the maximum and minimum disparity values in the sequence. Since the training of our DepthCrafter only involves the U-Net model, with the VAE freezing, we can pre-process the latents of videos and the corresponding depth sequences in advance. This caching mechanism significantly reduces the training time and memory consumption, as the latents do not need to be re-computed during the training process, and the VAE does not need to be loaded into the memory.

A.2. Training Details

We follow the EDM-framework [31] to train our DepthCrafter. This can be mathematically formulated as Eqs. (1) to (3). The c_{in}, c_{out}, c_{skip}, and c_{noise} in Eq. (3) are the EDM preconditionining functions [31, 53]:

$$\begin{aligned} c_{\text{in}}(\sigma_t) &= 1/\sqrt{1+\sigma_t^2}, \\ c_{\text{out}}(\sigma_t) &= -\sigma_t/\sqrt{1+\sigma_t^2}, \\ c_{\text{skip}}(\sigma_t) &= 1/(1+\sigma_t^2), \\ c_{\text{noise}}(\sigma_t) &= 0.25 \cdot \log(\sigma_t). \end{aligned} \tag{5}$$

c_{in} and c_{out} are used to scale the input and output magnitudes, c_{skip} is used to modulate the skip connection, and c_{noise} is used to map the noise level σ_t into a conditioning input for the denoiser F_θ. The λ_{σ_t} in Eq. (2) effectively incurs a per-sample loss weight for balancing different noise levels, which is set as:

$$\lambda_{\sigma_t} = 1/c_{\text{out}}(\sigma_t)^2. \tag{6}$$

During training, we randomly sample the noise level σ_t from a log-normal distribution:

$$\ln(\sigma_t) \sim \mathcal{N}(0.7, 1.6^2), \tag{7}$$

which is following the EDM-framework [31] to target the training efforts to the relevant range.

We train our DepthCrafter on eight NVIDIA A100 GPUs with a learning rate of 10^{-5}, and a batch size of 8. We adopted the DeepSpeed ZeRO-2 strategy, gradient checkpointing, and mixed precision training to reduce memory consumption during training. We also highly optimize the U-Net structure and cache the latents to further reduce memory consumption. The first and third training stages consume around 40GB of GPU memory per device, while the second stage consumes around 80GB. The "temporal layers" mentioned in Sec. 3 are the layers performed on the time axis, such as temporal transformer and temporal resnet blocks. The remaining layers are spatial layers, such as spatial transformers and spatial resnet blocks.

A.3. Benchmark Evaluation Details

Since existing monocular depth estimation methods and benchmarks are mainly tailored for static images, we recompile the public benchmarks to evaluate the video depth estimation methods. First, we re-format the testing datasets in the form of videos that are originally in the form of images. Specifically, for the ScanNet V2 dataset [12], we extracted the first 90 RGB-D frames from the original sensor data sequences at a rate of 15 frames per second. Besides, there are black regions in the corners of the images due to the camera calibration, which would affect the depth estimation evaluation. We carefully crop the borders of the images to remove the black regions, *i.e.* cropping 8 pixels from the top and bottom, and 11 pixels from the left and right. For the KITTI dataset [19], we extracted the first 110 frames from the original sequences without downsampling the frame rate, since the difference between consecutive frames is relatively large. And for the Bonn dataset [44], which was usually not included in the evaluation due to the small scale, but we find it is a good complement to the ScanNet dataset for indoor scenes as it contains dynamic contents while ScanNet contains only static scenes. We selected five sequences from the Bonn dataset, each with 110 frames, for evaluation. For the Sintel dataset [7], as it is a synthetic dataset, we directly used the original sequences.

For the evaluation metrics, we followed the insight from the commonly used metrics in the depth estimation, includ-

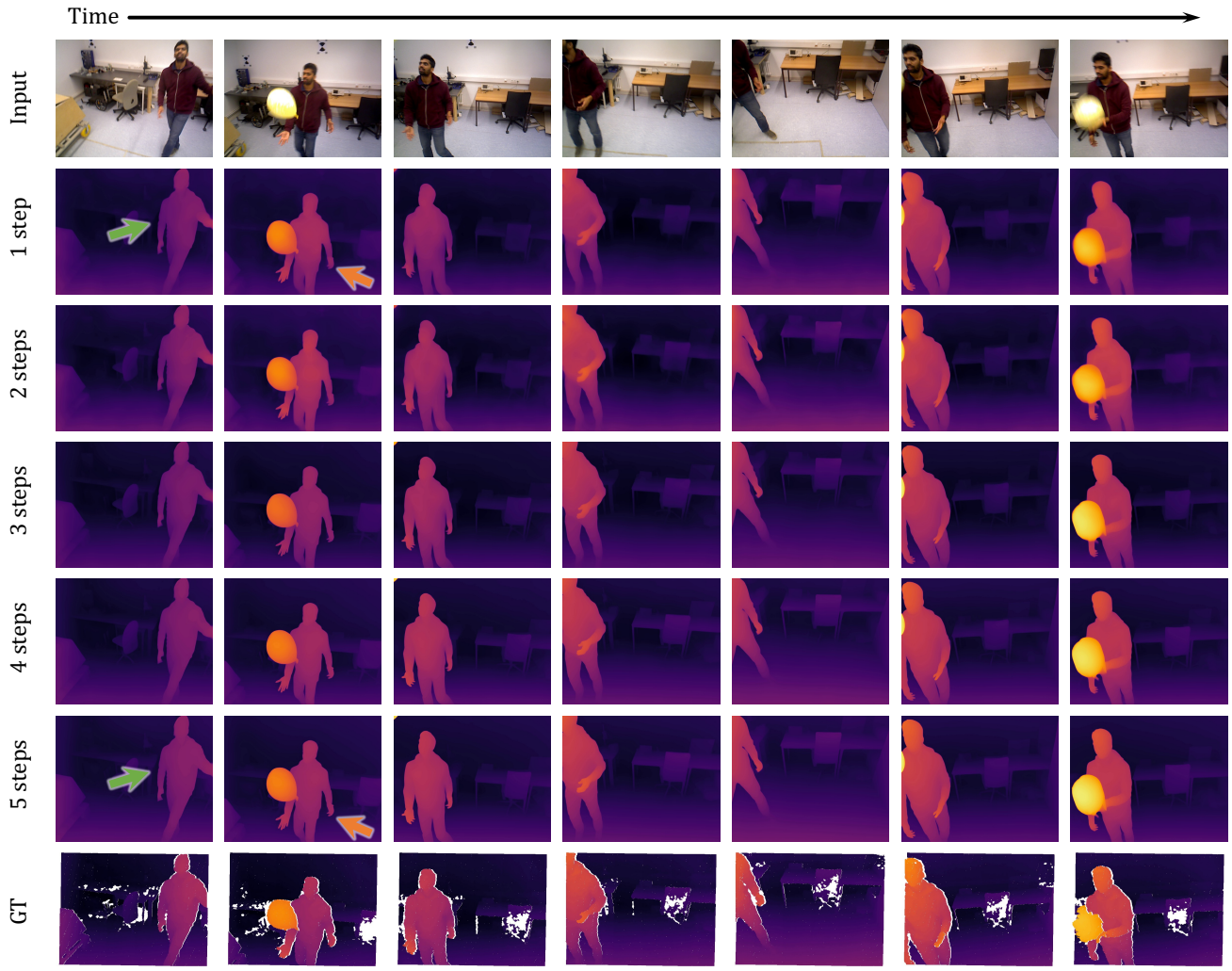

Figure S1. Effects of the number of denoising steps in our DepthCrafter. We show an example from the Bonn dataset [44], where the depth sequences are generated with different numbers of denoising steps. The green and orange arrows indicate the regions where more denoising steps can refine the depth details.

Table S1. Performance comparison of our DepthCrafter with different numbers of denoising steps. For reference, we also include the results of Marigold [32] and Depth-Anything-V2 [68]. The inference speed is measured in milliseconds per frame at the resolution of 1024×576. **Best** and second best results are highlighted.

Method	**Steps**	***ms* / frame** ↓	**Sintel** (∼50 frames)		**Scannet** (90 frames)		**KITTI** (110 frames)		**Bonn** (110 frames)	
		@1024×576	**AbsRel**↓	δ_1↑	**AbsRel**↓	δ_1↑	**AbsRel**↓	δ_1↑	**AbsRel**↓	δ_1↑
Marigold [32]		1070.29	0.532	0.515	0.166	0.769	0.149	0.796	0.091	0.931
Depth-Anything-V2 [68]		**180.46**	0.367	0.554	0.135	0.822	0.140	0.804	0.106	0.921
DepthCrafter (Ours)	1	337.10	0.319	0.651	0.132	0.826	0.138	0.812	0.084	0.954
	2	369.28	0.301	0.661	0.132	0.828	0.138	0.814	0.083	0.955
	3	401.47	0.273	0.693	**0.123**	0.854	0.111	0.877	0.073	0.971
	4	433.65	0.293	**0.697**	**0.123**	**0.856**	0.107	0.888	0.072	0.971
	5	465.84	**0.270**	**0.697**	**0.123**	**0.856**	**0.104**	**0.896**	**0.071**	0.972
	6	498.03	0.299	0.696	0.124	0.851	0.105	0.891	0.072	**0.973**
	10	626.72	0.291	0.694	0.125	0.849	0.106	0.890	0.073	0.972
	25	1109.42	0.292	**0.697**	0.125	0.848	0.110	0.881	0.075	0.971

Table S2. Effectiveness of our three-stage training strategy. We show the performance of our DepthCrafter with different training stages on the Sintel, Scannet, KITTI, and Bonn datasets. For reference, we also include the results of Marigold [32] and Depth-Anything-V2 [68]. **Best** and second best results are highlighted.

Method	Training Stages	**Sintel** (∼50 frames)		**Scannet** (90 frames)		**KITTI** (110 frames)		**Bonn** (110 frames)	
		AbsRel↓	δ_1↑	**AbsRel**↓	δ_1↑	**AbsRel**↓	δ_1↑	**AbsRel**↓	δ_1↑
Marigold [32]		0.532	0.515	0.166	0.769	0.149	0.796	0.091	0.931
Depth-Anything-V2 [68]		0.367	0.554	0.135	0.822	0.140	0.804	0.106	0.921
DepthCrafter (Ours)	1	0.322	0.626	0.170	0.721	0.174	0.724	0.103	0.917
	2	0.316	0.675	0.134	0.826	0.127	0.844	0.090	0.935
	3	**0.270**	**0.697**	**0.123**	**0.856**	**0.104**	**0.896**	**0.071**	**0.972**

ing the absolute relative error (AbsRel) and the δ_1 metric, but modified the scale and shift alignment from per-image to per-video. This is because the depth values for a video should be consistent across frames, otherwise, the depth sequences would be flickering. During evaluation, we first align the depth sequences to the ground truth by the scale and shift, using a least-square optimization. Following MiDas [49], we cap the maximum depth values to a certain value for different datasets, *e.g.*, 70 meters for the SinTel dataset, 80 meters for the KITTI dataset, and 10 meters for the ScanNet, Bonn, and NYUv2 datasets.

B. Additional Evaluations

B.1. Effect of Number of Denoising Steps

During inference, the number of denoising steps is a crucial hyperparameter that affects the trade-off between the inference speed and the depth estimation quality. The practice in image-to-video diffusion models [3] is to set the number of denoising steps to around 25. However, as shown in Fig. S1, we find that the number of denoising steps can be reduced significantly for video depth estimation, even one step works well. This is because the video depth estimation task is more deterministic than the video generation task. And we can see in the figure that more denoising steps would consistently improve the structure details of the generated depth sequences. In Tab. S1, we show the results of our DepthCrafter with different numbers of denoising steps. We can see that our DepthCrafter significantly outperforms existing strong baselines, such as Marigold [32] and Depth-Anything-V2 [68], even with only one denoising step. The performance of our DepthCrafter is increased with more denoising steps, but the improvement gets saturated after five steps. Thus we set the number of denoising steps to five in our experiments, which achieves a good trade-off between the inference speed and the depth estimation quality. The inference speed of our DepthCrafter with five denoising steps is 465.84 ms per 1024×576 frame, which is acceptable for many applications.

B.2. Effectiveness of Training Stages

In the main paper, we ablate the performance of our DepthCrafter with three training stages, only on the Sintel [7] dataset. To complement the evaluation, we further evaluate the effectiveness of our three-stage training strategy on all the datasets, including Sintel [7], Scannet [12], KITTI [19], and Bonn [44]. As shown in Tab. S2, we can observe that, even only with the first two stages, our DepthCrafter already outperforms the existing strong baselines, such as Marigold [32] and Depth-Anything-V2 [68]. More importantly, the performance improvement with the training stages is consistent across all the datasets. It indicates that our three-stage training strategy is effective for improving the generalization ability of our DepthCrafter to diverse open-world videos.

B.3. Effects of Classifier-Free Guidance

Classifier-free guidance (CFG) is proven to be effective in improving the details of the generated videos in video diffusion models [3, 8, 9, 64, 65]. In our DepthCrafter, we also investigate the effectiveness of CFG in video depth estimation. As shown in Fig. S2, we show an example frame from the KITTI dataset, where the results of our DepthCrafter with and without CFG are compared. We can see that the CFG can indeed improve the visual details of the generated depth sequences, especially for the fine-grained structures. However, we find that the CFG may slightly degrade the quantitative accuracy of the depth estimation, as shown in Tab. S3. This may be because the CFG is designed for improving the details of the generated videos, while the depth estimation task is more deterministic and requires more accurate predictions. Since adopting the CFG would also introduce additional computation, we do not use the CFG in our DepthCrafter for the main experiments. However, if the users are more interested in the visual details of the depth sequences, they can consider incorporating the CFG into our DepthCrafter.

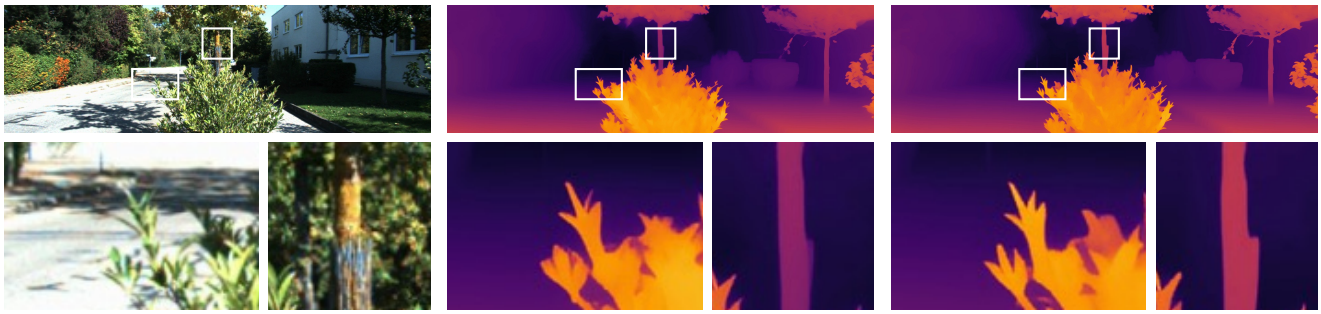

Figure S2. Effects of the classifier-free guidance (CFG) on the generated depth sequences. For better visualization, we blow up two regions that contain structures with fine-grained details.

Table S3. Effects of the classifier-free guidance (CFG). For reference, we also include the results of Marigold [32] and Depth-Anything-V2 [68]. **Best** and second best results are highlighted.

Method	**Sintel** (∼50 frames)		**Scannet** (90 frames)		**KITTI** (110 frames)		**Bonn** (110 frames)	
	AbsRel↓	δ_1↑	**AbsRel**↓	δ_1↑	**AbsRel**↓	δ_1↑	**AbsRel**↓	δ_1↑
Marigold [32]	0.532	0.515	0.166	0.769	0.149	0.796	0.091	0.931
Depth-Anything-V2 [68]	0.367	0.554	0.135	0.822	0.140	0.804	0.106	0.921
DepthCrafter W/O CFG	**0.270**	**0.697**	**0.123**	**0.856**	**0.104**	**0.896**	**0.071**	**0.972**
DepthCrafter W/ CFG	0.315	0.692	**0.123**	0.850	0.108	0.885	0.076	**0.972**